Developing and Deploying the Xena Gene Set Enrichment Analysis Appyter

Cally Lin

UCSC Genomics Institute, Xena Cancer Browser.

Abstract

UCSC Xena is a web-based visual integration and exploration tool for multiomic data and associated clinical and phenotypic annotations. It showcases seminal cancer genomics datasets from TCGA, the Pan-Cancer Atlas, GDC, PCAWG, ICGC, and more; a total of more than 1500 datasets across 50 cancer types. Researchers can easily view and explore public data, their own private data, or both using the Xena Browser. Xena users want to be able to run Gene Set Enrichment Analysis (GSEA) on the subgroups they created in Xena. To enable this functionality for users, a new appyter tool that runs blitzGSEA [3] was created using the Appyter framework developed by the Ma'ayan lab [1].Traditional GSEA is the industry standard in bioinformatics, but it was too computationally-intensive to run in the browser. blitzGSEA was evaluated and found to have better performance than the traditional GSEA implementation while producing similar, biologically consistent results. The appyter was deployed on the production Xena Browser using Docker and AWS EC2.

Keywords: Xena, gene set enrichment analysis, gene set, enrichment score, Appyter, Docker, AWS

1 Introduction

Gene set enrichment analysis (GSEA) is a computational method commonly used by researchers and scientists to determine whether there is an association between a set of genes and disease phenotype [5]. GSEA-P, GSEApy, and fGSEA are all different GSEA implementations. A group of researchers at Ma'ayan Labs developed blitzGSEA recently, and the algorithm is claimed to have a "significant improvement in performance." [3] Before implementing blitzGSEA into Xena for users to run, the results produced must be confirmed to make biological sense, and the runtime and memory usage of the algorithm is established to be efficient. Thus, the functionalities of blitzGSEA were tested and run with several files; each analysis was completed efficiently and produced results that were logical when compared to those calculated through traditional GSEA. With this validation, Xena's previous success with differential gene expression analysis was built upon, using the Appyter framework [1] to create a new appyter tool that runs blitzGSEA. The blitzGSEA appyter was tested and reviewed to make sure it is accessible and functional, and it was deployed on the production Xena Browser.

2 Methods

2.1 Testing blitzGSEA

The example written by the Ma'ayan Lab was run to understand what functionalities blitzGSEA has. It was found that the Pandas data frame returned could be used to provide users with a full table result in a `.tsv/.csv` format which includes the enrichment score (ES), normalized enrichment score (NES), and other data for every gene set of their library. In addition, the functions written within the source code could generate a top table and running sum plot for a gene set of the user's choosing.

With this information, I wrote a script [1] that combines the functions in blitzGSEA to test whether the results were biologically sound. It takes in two files, `.gmt` and `.rnk` (a gene set library and signature), and reformats them to the proper form for blitzGSEA to run. Then, it generates a full-table and detailed results

My mentor, Dr. Zhu, gave me 2 differential gene expression signatures and 3 gene set libraries with traditional GSEA results. I ran 6 analyses using blitzGSEA and collected the runtime, memory usage, and results for each. Several scatter plots were graphed between the NES values using Tableau (a data visualization tool) to verify the validity of the values produced by blitzGSEA.

2.2 The blitzGSEA appyter

Appyter is a software framework developed by researchers at the Ma'ayan lab. It allows users to run bioinformatic methods without having to interact with code directly. Users can upload or fetch data and configure tweakable parameters on a web-based form; the variables inputted by the user on the form are inserted into a Jupyter Notebook, and the Notebook is executed. A Jupyter Notebook with the results is rendered for the user. The Notebook consists of markdown text and interactive/static plots along with the options to download the Notebook (in `.ipynb`, `.html`, or `.zip` format), view the source code, or run the Appyter locally.

The web-form for the blitzGSEA appyter was developed modeling the Xena Differential Gene Expression appyter [6]. Many analyses were kept except those involved with Enrichr and L1000 small molecule analysis. A section for selecting gene set libraries and the blitzGSEA parameters were added. The modified code for blitzGSEA [4] was implemented in addition to the DE appyter code with some changes to the production of the full table, top table, running sum plot, and detailed output.

The source code for blitzGSEA [4] produces a full table, top table, and a detailed output and running sum plot for one gene set. However, the interactivity of each is limited, as they are outputted in a static `.tsv`/`.csv` and `.png` format. To allow for better visualization and functionalities, I utilized Dash [4], a Python framework that enables the creation of interactive data visualization web apps, to create a dashboard separate from the notebook to compute the top table, running sum plots, and detailed outputs. This allows users to access more than one gene set's running sum plot and detailed output. The dashboard was intended to be in the Notebook so users did not have to open a separate tab; however, upon further investigation and reaching out to one of the developers of Appyter (Daniel Clarke), it was found that due to the design of Appyter, it isn't possible for end-users to send information to the kernel, as it would affect execution. This is why the dashboard is opened in a separate tab and the full table in the Notebook isn't interactive. In the dashboard, users are given the option to download the figures and table to their machine. I also added a circular spinner for each figure to display when loading, so users know that it will appear momentarily.

2.3 Containerizing the application using Docker and pushing the image to Amazon Elastic Container Registry before deployment

A new container image was built for the blitzGSEA appyter using Docker on an EC2 server. This process began with validating the appyter with a script written by the Ma'ayan lab to check that the repository had the necessary files, such as `requirements.txt`, and in the correct format. The `.env` file was adapted to be under the appropriate Docker registry and use the latest version of Appyter. A Dockerfile was constructed for the appyter, and using Node.js, the app was built according to appyter.json, a file containing information about the blitzGSEA appyter. A Compose file was created, and the Docker images were built for the appyter and app with the Dockerfiles. The image was run as a container using the command `docker run -p 5000:5000 image_id`, and the Xena Browser was run on localhost after line 715 [5] was changed to connect the browser to the virtual server the appyter is on.

A test was done with a sample data set, but after the form was submitted in the appyter, the notebook would not start and produced an error. I went through the source code for Appyter's catalog and found documentation on how to run an appyter with Docker [2]. Using the command `docker run --device /dev/fuse --cap-add SYS_ADMIN --security-opt apparmor:unconfined -p 5000:5000 image_id` instead, the container was started, and the notebook was able to execute without error. The previous

Table 1 Runtime and memory usage running blitzGSEA — The user runtime (in seconds) and memory usage (in megabytes) of running blitzGSEA with a specific signature and geneset library file. The correlation coefficient, r, is obtained through the comparison of NES values blitzGSEA vs. traditional GSEA.

Signature (.rnk)	**Geneset (.gmt)**	**Size**	**Time (s)**	**Memory (mb)**	r
basal_versus_lum	c2_c3	7494	121.28	303.968	0.78891888
basal_versus_lum	c5.all.v7.5.1.symbols	15473	119.78	477.532	—
basal_versus_lum	msigdb.v7.5.1.symbols	32880	144.36	994.368	—
ESR1_low_versus_ESR1_high	c2_c3	7494	116.80	308.456	0.88312513
ESR1_low_versus_ESR1_high	c5.all.v7.5.1.symbols	15473	118.71	444.304	0.87125943
ESR1_low_versus_ESR1_high	msigdb.v7.5.1.symbols	32880	144.95	939.568	0.88575674

container was stopped, and the blitzGSEA Docker image was run again in detached mode and bind mounted to a data directory with the addition of `-d` and `-v` in the command; this allows for the container to run in the background and persist data.

The functionalities of the blitzGSEA appyter were tested to be sure that all the features developed worked as they should. Dr. Zhu, helped set up the Amazon ECR after confirming the appyter runs successfully, and the blitzGSEA appyter image was pushed to Amazon ECR.

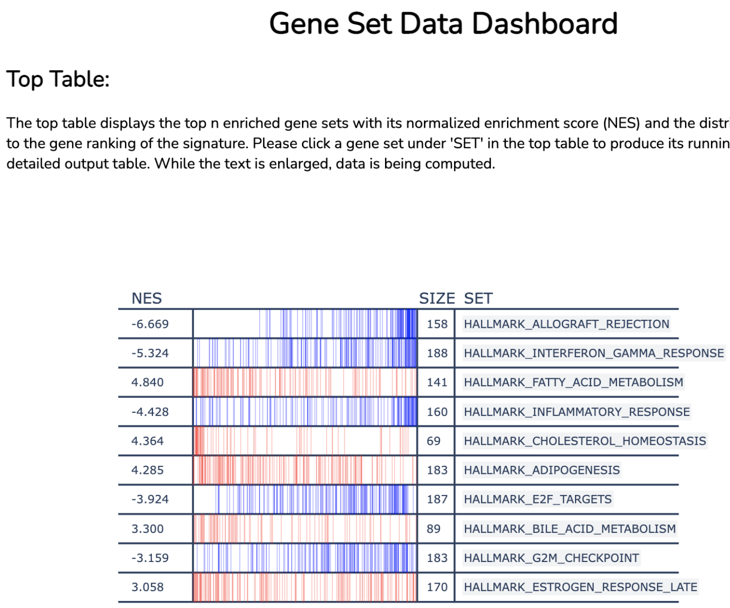
3 Results

3.1 Traditional GSEA vs. blitzGSEA

Each analysis was completed within a reasonable amount of time (user time $<$ 150 seconds) and memory usage ($<$ 950 megabytes) as seen in Table 1. The scatter plot between all positive and negative NES' from the analyses for both blitzGSEA and traditional GSEA had a correlation coefficient (r) $>$ than 0.78 which is an indication of high accuracy.

3.2 The blitzGSEA appyter

The blitzGSEA appyter will be available for researchers and scientists to use on the Xena Browser. After selecting a study to explore and two different variables, users will be able to select `GSEA` from the menu of the categorical variable. A form will open up, and the gene expression data and the two subgroups selected through Xena will be loaded into the appropriate sections. Users will select a gene set library retrieved from the Molecular Signature Database (MSigDB) and have the option to alter other parameters. When they submit the form, a notebook will open. If the data has not been previously run before with the same parameters, users will be able to see the cells of the notebook execute; otherwise, a copy of the previous results will load. When the notebook has been successfully completed, a Dash app can be accessed to view the interactive top table of their top `n` gene sets, and upon clicking a gene set's name, their running sum plot and detailed output will generate (Figure 1). Users can save the top table and running sum plot as a `.png` and the detailed output as a `.csv`. The notebook can be downloaded as a `.ipynb`,


Gene Set Data Dashboard

Top Table:

The top table displays the top n enriched gene sets with its normalized enrichment score (NES) and the distribution of hits relative to the gene ranking of the signature. Please click a gene set under 'SET' in the top table to produce its running sum plot and detailed output table. While the text is enlarged, data is being computed.

NES		SIZE	SET
-6.669		158	HALLMARK_ALLOGRAFT_REJECTION
-5.324		188	HALLMARK_INTERFERON_GAMMA_RESPONSE
4.840		141	HALLMARK_FATTY_ACID_METABOLISM
-4.428		160	HALLMARK_INFLAMMATORY_RESPONSE
4.364		69	HALLMARK_CHOLESTEROL_HOMEOSTASIS
4.285		183	HALLMARK_ADIPOGENESIS
-3.924		187	HALLMARK_E2F_TARGETS
3.300		89	HALLMARK_BILE_ACID_METABOLISM
-3.159		183	HALLMARK_G2M_CHECKPOINT
3.058		170	HALLMARK_ESTROGEN_RESPONSE_LATE

Fig. 1 The Dash dashboard — An example interactive top table showing the top n gene sets.

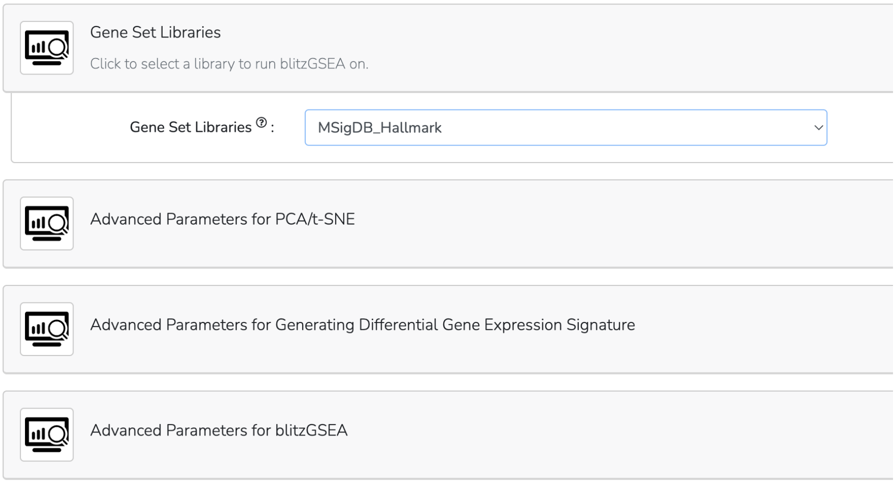
`.html`, or `.zip`, and the code behind the notebook can be viewed by clicking "Toggle Code."

4 Discussion

Over the year of completing this project, many obstacles were faced and changes were made to the blitzGSEA appyter code, Appyter source code, and blitzGSEA source code.

When the project first began and research and testing were being done on blitzGSEA's accuracy and efficiency, it was found that the way calculations were being done was slightly off. It occasionally produced *-inf* and *inf* values for enrichment scores which were invalid and did not reflect the values from traditional GSEA. It is hypothesized that this was due to the way Numpy equations were organized and values becoming too close to zero. I reorganized the Numpy equations (line 280 and 292) in the source code, and the miscalculations were fixed. The incorrect data became an issue again during deployment due to an update blitzGSEA made in their source code. The incorporation of line 228 was adding too much noise to signature values, resulting in enrichment scores being extremely off, so I commented out the line of code. This fixed the inaccurate values, and the altered blitzGSEA source code is what the blitzGSEA appyter uses to run GSEA [4].

Additionally, to avoid overwhelming users with too many options that the typical user will not need to change, the parameter setting sections for each



Gene Set Libraries

Click to select a library to run blitzGSEA on.

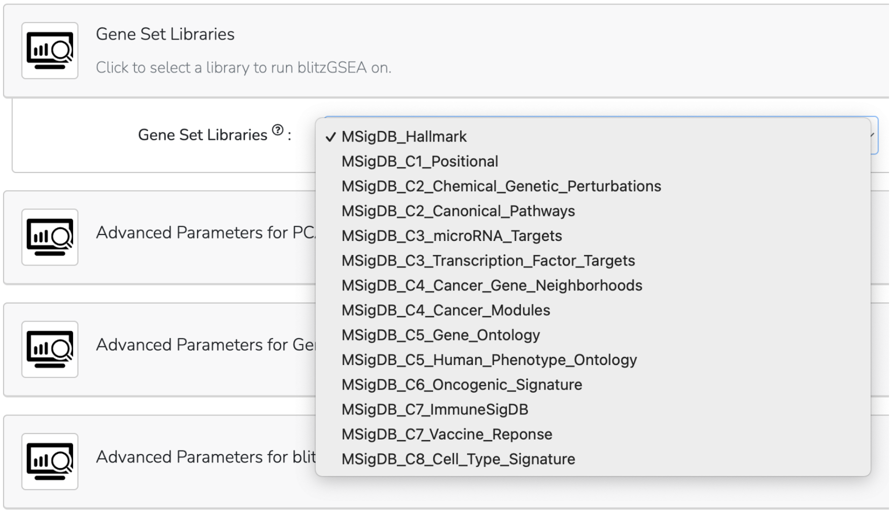
Gene Set Libraries ⓘ: MSigDB_Hallmark

Advanced Parameters for PCA/t-SNE

Advanced Parameters for Generating Differential Gene Expression Signature

Advanced Parameters for blitzGSEA

Fig. 2 The form loaded upon clicking "GSEA" from Xena — The gene set library selection is open whereas the advanced parameters sections are collapsed



Gene Set Libraries

Click to select a library to run blitzGSEA on.

Gene Set Libraries ⓘ:

✓ MSigDB_Hallmark
MSigDB_C1_Positional
MSigDB_C2_Chemical_Genetic_Perturbations
MSigDB_C2_Canonical_Pathways
MSigDB_C3_microRNA_Targets
MSigDB_C3_Transcription_Factor_Targets
MSigDB_C4_Cancer_Gene_Neighborhoods
MSigDB_C4_Cancer_Modules
MSigDB_C5_Gene_Ontology
MSigDB_C5_Human_Phenotype_Ontology
MSigDB_C6_Oncogenic_Signature
MSigDB_C7_ImmuneSigDB
MSigDB_C7_Vaccine_Reponse
MSigDB_C8_Cell_Type_Signature

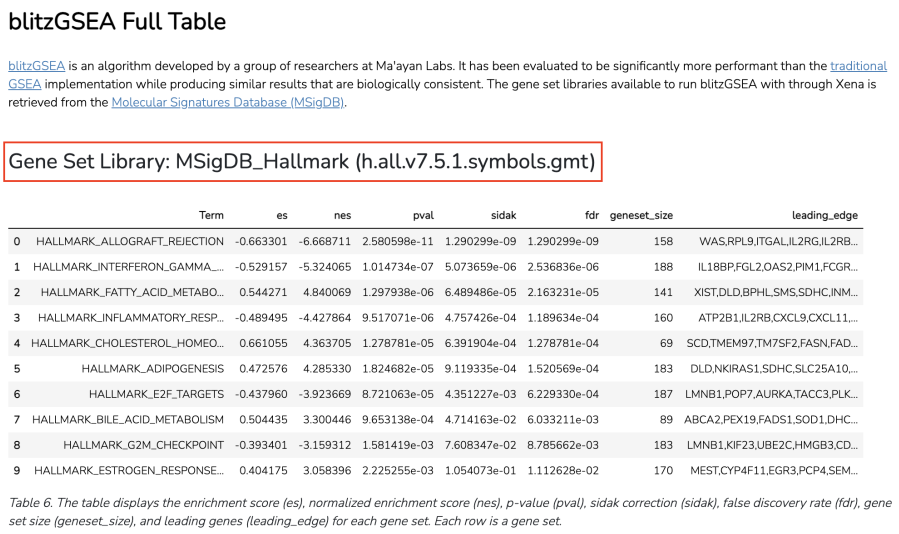
Advanced Parameters for PC

Advanced Parameters for Ge

Advanced Parameters for blit

Fig. 3 The drop-down menu for selecting a gene set library — The gene set library selection only allows users to choose one library.

analysis (PCA/t-SNE, Differential Gene Expression, and blitzGSEA) were set to collapse upon loading of the form (Figure 2). This was originally done by modeling the Xena Differential Gene Expression [6] appyter, but it did not work as intended. After further investigation, it was found that there were several code changes that had been made to the Appyter source code, which changed how Appyter functioned. Rather than including a template file that is read and implemented during the making of the form, the new code



blitzGSEA Full Table

blitzGSEA is an algorithm developed by a group of researchers at Ma'ayan Labs. It has been evaluated to be significantly more performant than the traditional GSEA implementation while producing similar results that are biologically consistent. The gene set libraries available to run blitzGSEA with through Xena is retrieved from the Molecular Signatures Database (MSigDB).

Gene Set Library: MSigDB_Hallmark (h.all.v7.5.1.symbols.gmt)

	Term	es	nes	pval	sidak	fdr	geneset_size	leading_edge
0	HALLMARK_ALLOGRAFT_REJECTION	-0.663301	-6.668711	2.580598e-11	1.290299e-09	1.290299e-09	158	WAS,RPL9,ITGAL,IL2RG,IL2RB...
1	HALLMARK_INTERFERON_GAMMA_...	-0.529157	-5.324065	1.014734e-07	5.073659e-06	2.536836e-06	188	IL18BP,FGL2,OAS2,PIM1,FCGR...
2	HALLMARK_FATTY_ACID_METABO...	0.544271	4.840069	1.297938e-06	6.489486e-05	2.163231e-05	141	XIST,DLD,BPHL,SMS,SDHC,INM...
3	HALLMARK_INFLAMMATORY_RESP...	-0.489495	-4.427864	9.517071e-06	4.757426e-04	1.189634e-04	160	ATP2B1,IL2RB,CXCL9,CXCL11,...
4	HALLMARK_CHOLESTEROL_HOMEO...	0.661055	4.363705	1.278781e-05	6.391904e-04	1.278781e-04	69	SCD,TMEM97,TM7SF2,FASN,FAD...
5	HALLMARK_ADIPOGENESIS	0.472576	4.285330	1.824682e-05	9.119335e-04	1.520569e-04	183	DLD,NKIRAS1,SDHC,SLC25A10,...
6	HALLMARK_E2F_TARGETS	-0.437960	-3.923669	8.721063e-05	4.351227e-03	6.229330e-04	187	LMNB1,POP7,AURKA,TACC3,PLK...
7	HALLMARK_BILE_ACID_METABOLISM	0.504435	3.300446	9.653138e-04	4.714163e-02	6.033211e-03	89	ABCA2,PEX19,FADS1,SOD1,DHC...
8	HALLMARK_G2M_CHECKPOINT	-0.393401	-3.159312	1.581419e-03	7.608347e-02	8.785662e-03	183	LMNB1,KIF23,UBE2C,HMGB3,CD...
9	HALLMARK_ESTROGEN_RESPONSE...	0.404175	3.058396	2.225255e-03	1.054073e-01	1.112628e-02	170	MEST,CYP4F11,EGR3,PCP4,SEM...

Table 6. The table displays the enrichment score (es), normalized enrichment score (nes), p-value (pval), sidak correction (sidak), false discovery rate (fdr), gene set size (geneset_size), and leading genes (leading_edge) for each gene set. Each row is a gene set.

Fig. 4 The blitzGSEA full table section — The label for the full table includes the gene set library name seen from the form and the .gmt file name.

creates the form in the source code and implements the collapse feature that is already available in JavaScript. The minified file, form.js, in the biojupies profile directory of the Appyter source code was unminified using Unminify. I changed line 421 to check whether the variable "collapse" exists, and if it does, the section will be collapsed. The forked repository is, again, what the blitzGSEA appyter uses [4].

The blitzGSEA appyter was initially designed to allow users to select more than one gene set library to run blitzGSEA on at a time, but this became a problem towards the end of the project. This is because multiple links could not be created for the Dash app due to the way the Appyter source code reads the blueprints directory and applies it to the appyter. Hence, my mentors and I decided that users would be limited to one gene set library selection, so they would not be confused as to why only the last gene set library they selected has a Dash app. As seen in Figure 3, the gene set library selection is a drop-down menu instead of a multi-selection menu like before.

Other changes were implemented, such as better labeling and descriptions, to make the appyter more user-friendly and maximize convenience. This included adding the selected gene set library name from the form to the full table (Figure 4) along with the file name itself to prevent confusion when downloading the table as a .csv file. A description for the dashboard with the top table, running sum plot, and detailed output was added with its own section too, so users can better understand what will happen when they click on the link.

Aside from the final changes made to the blitzGSEA appyter, the Xena Browser source code was changed to include a menu option (as seen in Figure 5)

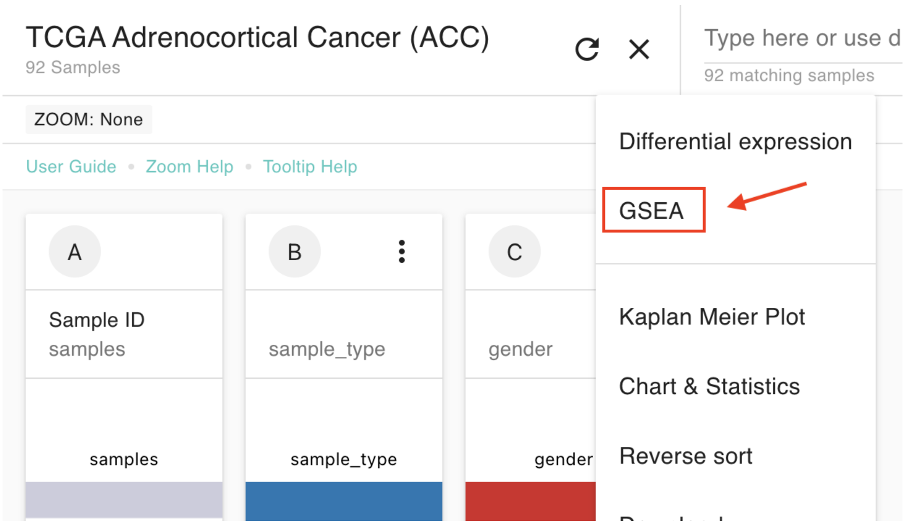


Fig. 5 The column menu on the Xena Browser — The new column menu in the Xena Browser with the option to run the GSEA analysis. This option is only available for categorical data.

to run GSEA with data from Xena. The development Xena Browser was connected to the appyter running on AWS, and the quality assurance team at the Genomics Institute is testing the appyter to make sure there are no issues.

5 Conclusion

Overall, the project was a success, and the blitzGSEA appyter is expected to be released at the end of March 2023. In the future, if possible, the feature allowing users to run multiple gene set libraries at once and producing an interactive Dash app for each will be added. The option to use your own gene set library instead of the provided MSigDB gene set libraries will be implemented as well.

Acknowledgments

I would like to thank my wonderful mentors, Jingchun Zhu and Mary Goldman, at the UCSC Genomics Institute, as this work would not have been made possible without their continuous support and guidance. I would also like to thank CZI's Essential Open Source Software for Science Diversity and Inclusion Grants for the funding.

Code Availability

References

[1] callylin (June 3, 2022). xena_blitzGSEA https://github.com/callylin/xena_blitzGSEA

[2] Maayan Lab (March 13, 2023). appyter-catalog: A catalog of appyters. https://github.com/MaayanLab/appyter-catalog

[3] Maayan Lab (March 16, 2023). Appyter. https://github.com/callylin/appyter.git - Forked repository

[4] Maayan Lab (March 13, 2023). blitzGSEA. https://github.com/callylin/blitzgsea.git - Forked repository

[5] ucscXena (March 8, 2023). UCSC Xena Client https://github.com/ucscXena/ucsc-xena-client

[6] ucscXena (August 5, 2021). Xena Gene Differential Gene Expression analysis pipeline. https://github.com/ucscXena/Xena_DE_Analysis_Pipeline.git

References

[1] Clarke, D., Jeon, M., Stein, D. J., Moiseyev, N., Kropiwnicki, E., Dai, C., Xie, Z., Wojciechowicz, M. L., Litz, S., Hom, J., Evangelista, J. E., Goldman, L., Zhang, S., Yoon, C., Ahamed, T., Bhuiyan, S., Cheng, M., Karam, J., Jagodnik, K. M., Shu, I., … Ma'ayan, A. (2021). Appyters: Turning Jupyter Notebooks into data-driven web apps. Patterns (New York, N.Y.), 2(3), 100213. https://doi.org/10.1016/j.patter.2021.100213

[2] Docker Overview. Docker Documentation. (2023, January 13). Retrieved January 16, 2023, from https://docs.docker.com/get-started/overview/

[3] Lachmann, A., Xie, Z., & Ma'ayan, A. (2022). blitzGSEA: Efficient computation of Gene Set Enrichment Analysis through Gamma distribution approximation. Bioinformatics (Oxford, England), btac076. Advance online publication. https://doi.org/10.1093/bioinformatics/btac076

[4] Plotly. (n.d.). Introduction to Dash. Plotly. https://dash.plotly.com/introduction

[5] Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., Paulovich, A., Pomeroy, S. L., Golub, T. R., Lander, E. S., & Mesirov, J. P. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proceedings of the National Academy of Sciences of the United States of America, 102(43), 15545–15550. https://doi.org/10.1073/pnas.0506580102